Google and the Mathematics of Web Search

Michael Rempe
Whitworth University, Spokane, WA

Michael Rempe (B.S., University of Colorado; Ph.D., Northwestern University) has been with the Department of Mathematics and Computer Science at Whitworth University since 2009. Michael studies Mathematical Neuroscience and enjoys involving students in his research. When not doing things that relate primarily to mathematics, Michael can be found cycling, hiking, snow-shoeing, or camping in the beautiful Northwest.

1. Background

1.1. Web Search Before Google

When I was a junior in high school I began to think about where I wanted to go to college. I decided to use the Internet to learn more about the schools I was interested in. However, I did not know how to find the homepages for the Universities. As this was 1994 and the Internet was relatively young, I did not know that one could type in something like `www.colorado.edu` to find the homepage of the University of Colorado, for instance.

My Father informed me that I could use AltaVista to search the Internet for the university web sites. When I would type the name of a university into AltaVista the search was slow and the results would be less than helpful. For instance, typing in the name of a university would often return results for particular student groups, departments, or announcements at that university, rather than the main university homepage. My technique was to click on one of these links and try to follow links from page to page until I reached the main university web page. This was a frustrating endeavor, however, and I found myself wondering why AltaVista didn't know that I wanted the main university webpage, instead of more specific, less important web pages that were affiliated with the university, but clearly had a much smaller intended audience.

But this raises an interesting question: how do we determine which web pages are more "important" than others? If AltaVista knew which web pages were more important than others, it could list my search results in the order of importance, with the most relevant pages at the top of the list.

In what follows I will describe how Google solves this problem of determining the relative importance of web pages. One of the algorithms Google uses to rank the results of searches is called PageRank. The main concept behind PageRank is that the connected structure of the internet gives information about which web pages are relatively important. Each web page is assigned its own PageRank score and when Google returns search results it sorts the list according to each page's PageRank score.

1.2. What Happens After You Type a Search into Google

PageRank is only one part of what Google does after a user enters some search terms. When search terms are entered into Google, that text is sent to the Google web server. The web server sends the query terms to an index server. The index server contains a large list telling which pages contain those particular search terms. After the index server has found a list of the appropriate pages, the query travels to document servers which retrieve the actual contents of the relevant web pages, not just their addresses. A small part of the content of each page is displayed when Google lists the results. Finally, before Google shows the list of relevant pages (along with snippets), the results are ordered such that the most relevant pages are at the top of the list. PageRank is part of this final step: ordering the search results, but it is not the only part. The other portions of the sorting algorithm are proprietary. A nice summary of the entire search process can be found in [3].

1.3. Connectivity Implies Relative Importance

A crucial component of the internet is not only the pages themselves, but also the connections between the pages, called hyperlinks. Mathematically speaking, the internet can be thought of as a large directed graph where the web pages are nodes, and the edges are hyperlinks.

Part of the brilliance of the PageRank algorithm is the idea that the connected structure of the web can tell us about relative importance of individual web pages regardless of the actual content in those pages. A helpful analogy to help us understand how connections can tell us about relative importance is to think of each link as a recommendation. If there is a link from page A to page B, it's as if page A is endorsing or recommending page B. If a particular page has a lot of "inlinks" (pages linking to it) this suggests that that particular page is important. And PageRank takes this into account. More inlinks to a particular page typically results in a higher ranking. However, on the internet, as in life, not every recommendation is of the same worth. A recommendation from a famous person is surely more useful than a lot of recommendations from obscure people. So, PageRank takes this into account too. A link to page A from an important page helps raise page A's rank more than a link from an obscure page. So page A's rank depends on the ranks of the pages that link to page A, not just the sheer number of inlinks it has. Also, the relative value of each inlink or recommendation a page has needs to be considered in terms of how many outlinks that page makes. For instance, if page A gets a recommendation from page B, but it turns out that page B has many outgoing links (recommendations) then the recommendation from page B is not worth as much as it would have been if page B only made a few recommendations. This is consistent with the letter of recommendation idea: a letter from a famous person is no longer as valuable if it turns out that the recommender actually writes hundreds of recommendations every year. This idea of thinking of links as letters of recommendations is nicely summarized in [4].

A concise way to summarize these ideas is to say that a webpage is important if it is pointed to by other important web pages. This is circular-sounding argument, but it turns out that it works and actually works well if it is formulated mathematically. This is the central thesis of the PageRank algorithm.

2. PageRank

To formulate the PageRank algorithm mathematically, we can begin with the following conceptual equation:

ranking of page i $= \sum \dfrac{\text{rankings of page } j}{\text{number of outlinks from page } j}$, where all the pages j link to page i.

Being a bit more precise with our notation, we can write out the PageRank idea in the following manner:

$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|}, \tag{1}$$

where $r(P_i)$ = PageRank of page P_i, B_{P_i} = the set of all pages linking to P_i, and $|P_j|$ = the number of outlinks from page P_j.

Let's illustrate this concept with the simple web structure shown in Figure 1.

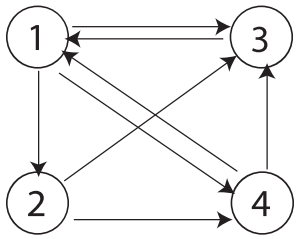

Figure 1: A simple connected web structure to illustrate PageRank

Writing out equations (1) for this particular system yields the following:

$$\begin{aligned} r_1 &= \frac{r_3}{1} + \frac{r_4}{2}, \\ r_2 &= \frac{r_1}{3}, \\ r_3 &= \frac{r_1}{3} + \frac{r_2}{2} + \frac{r_4}{2}, \\ r_4 &= \frac{r_1}{3} + \frac{r_2}{2}. \end{aligned}$$

This system of four equations can be written using matrix-vector notation:

$$\begin{pmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix},$$

which is simply an equation of the form $Ar = \lambda r$, a classic eigenvalue-eigenvector equation. So, the PageRank of each page in the sample web can be found by solving for the eigenvector corresponding to the eigenvalue 1. In this case

$$\mathbf{r} = \begin{pmatrix} 12 \\ 4 \\ 9 \\ 6 \end{pmatrix}.$$

If the eigenvector is normalized so its elements sum to 1 we are left with the following:

$$\mathbf{r} = \begin{pmatrix} 0.387 \\ 0.129 \\ 0.290 \\ 0.194 \end{pmatrix}.$$

So for this sample web, the most "important" page is page 1, followed by pages 3, 4 and 2. Note that the matrix $\mathbf{A}$ has entries between 0 and 1 and its columns sum to 1. Such matrices are said to be column-stochastic.

Another way to formulate this problem is to imagine a random web surfer who constantly follows links from one page to another [1]. Thinking of this as a stochastic process, the more "important" pages will be the ones that the random surfer spends most of his time visiting. This "random surfer" formulation is actually equivalent to the way the PageRank algorithm is set up above [4]. The PageRank of each page is the limiting probability that the surfer will be visiting that page at any particular time.

2.1. Existence and Uniqueness of Solutions

Mathematically speaking, this formulation raises a lot of questions, like how do we know that the matrix A above has an eigenvalue of 1? Also, how do we know that there is a unique eigenvector corresponding to the eigenvalue of 1, thereby giving a unique set of rankings for the pages in the web?

Much has been written on this topic, so here I will simply mention two of the important theorems. First of all it is easy to prove that every column stochastic matrix has 1 as an eigenvalue. And secondly, if the matrix $\mathbf{A}$ is positive and column-stochastic then the eigenspace for eigenvalue 1 has dimension 1. These theorems are nicely addressed in [2].

3. Problems in Real Webs

The theory developed above works well for simple webs like the one in Figure 1, but real networks of web pages are extremely large and they often have characteristics that cause fundamental problems with this approach. Here I will focus on two of the most common problems: dangling nodes and disconnected subwebs.

A "dangling node" refers to a web page that does not have any outgoing links. Thinking of the random surfer model, if the surfer makes it to this page, there will not be any links to follow to connect to other pages in the network. Since those other pages will not be visited, the random surfer won't get a clear picture of the connected structure of the web. This is quite common on the World Wide Web since many pages are documents or images that have inlinks, but don't have any outlinks. An example is shown in Figure 2.

In this case, if a random surfer follows a link to page 2, it will get stuck there. The corresponding

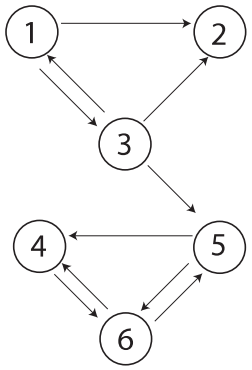

Figure 2: A simple web with a dangling node. If a random surfer gets to page 2, there isn't an outlink to connect it to any other page in the network.

connectivity matrix for this network is as follows:

$$\begin{pmatrix} 0 & 0 & 1/3 & 0 & 0 & 0 \\ 1/2 & 0 & 1/3 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 1/2 & 0 \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{pmatrix}.$$

Note that the second column contains all zeros due to the fact that page 2 doesn't have any outgoing links. To remedy this situation, the creators of PageRank proposed the following solution: if the surfer reaches a page that does not have any outlinks, the surfer chooses another page in the web randomly and transports there. This keeps the surfer from getting stuck on a dangling node and it changes the connectivity matrix as follows:

$$\begin{pmatrix} 0 & 1/6 & 1/3 & 0 & 0 & 0 \\ 1/2 & 1/6 & 1/3 & 0 & 0 & 0 \\ 1/2 & 1/6 & 0 & 0 & 0 & 0 \\ 0 & 1/6 & 0 & 0 & 1/2 & 1/2 \\ 0 & 1/6 & 1/3 & 0 & 0 & 1/2 \\ 0 & 1/6 & 0 & 1 & 1/2 & 0 \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{pmatrix}.$$

We will refer to this new matrix as $\mathbf{M}$. All entries in the second column are 1/6 since the surfer is equally likely to visit any page in the network if it is currently visiting page 2. After making this correction, the matrix $\mathbf{M}$ satisfies the criteria for the Perron-Frobenious Theorem and therefore has an eigenvalue of 1 with a unique corresponding eigenvector.

The second major problem that arises when applying this algorithm to the World Wide Web is that not every group of web pages is connected to every other group. Frequently there are "cliques" that form: groups of web pages that are connected to each other, but not to any other pages outside the clique. An example of a web with a disconnected subweb is shown in Figure 3, which is the same web as in Figure 2 with the connection from page 3 to 5 removed.

Continuing with the random surfer analogy, if the surfer were to start on page 1, it would be unable to reach pages 4, 5, or 6 simply by following links. This problem is again relatively

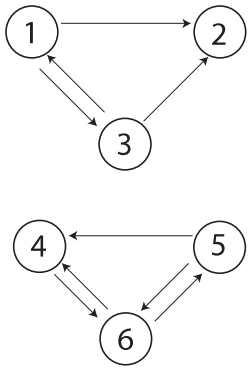

Figure 3: A simple network with a disconnected subweb.

easily overcome if instead of always following links, the random surfer occasionally teleports to another page in the network. To modify our linear equation to reflect this new strategy, we set up the matrix $\mathbf{G} = (1-\alpha)\mathbf{M} + \alpha\mathbf{S}$, where α is a constant between 0 and 1, and the matrix $\mathbf{S}$ contains the value $1/N$ in each position and N is the total number of pages in the web. The matrix $\mathbf{G}$ represents a balance between following the link structure of the web and randomly teleporting to another page. The parameter α can be chosen in such a way that the random surfer is mostly following links, and only occasionally teleporting to another site. PageRank is thought to use a value of $\alpha = 0.15$ [5].

For this particular example,

$$\mathbf{G} = (1-\alpha)\begin{pmatrix} 0 & 1/6 & 1/2 & 0 & 0 & 0 \\ 1/2 & 1/6 & 1/2 & 0 & 0 & 0 \\ 1/2 & 1/6 & 0 & 0 & 0 & 0 \\ 0 & 1/6 & 0 & 0 & 1/2 & 1/2 \\ 0 & 1/6 & 0 & 0 & 0 & 1/2 \\ 0 & 1/6 & 0 & 1 & 1/2 & 0 \end{pmatrix} + \alpha \begin{pmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{pmatrix}.$$

Now we can write

$$\mathbf{G}\,\mathbf{r} = \mathbf{r},$$

and $\mathbf{G}$ is now column stochastic, so using the theorems in Section 2.1 it can be proven that $\mathbf{G}\,\mathbf{r} = \mathbf{r}$ has a unique solution. Thus, a unique ranking of pages exists.

4. A Consequence of Ranking by Popularity: Google Bombs

Since the PageRank algorithm uses the connected structure of the internet to determine the relative importance of pages, the linking structure can be manipulated to alter the PageRank scores of a particular webpage. Called a *Google Bomb*, one of the most famous examples was in 2006 when the search term "miserable failure" listed the Biography of president George W. Bush as the first result. This happened because many bogus web pages were created that contained the keywords "miserable failure" and pointed to the biography of President Bush. Interestingly, MichaelMoore.com was the second hit, apparently the result of a Google Bomb created by Bush supporters. There have been many humorous examples of Google Bombs, although they are much less common now as it seems that Google has changed their ranking algorithm to be less prone to Google Bombs.

5. Google as a Teaching Tool for Linear Algebra

One of the most common questions I receive as a Mathematics instructor is "When are we ever going to use this?" One of the main reasons that I include a discussion and a homework assignment on PageRank is to answer that question. It helps the students make a real connection to something we all use every day.

I require my Linear Algebra students to complete one assignment related to PageRank. It consists of computing the PageRank of each website in a few small webs, some of which have dangling nodes and/or disconnected subwebs. I have my students use MATLAB and I spend most of one class period teaching them the basics. I also provide some sample code that they can modify instead of writing their own code from scratch.

Since the internet is so large, PageRank cannot rely on something like MATLAB's "eig" command to compute the eigenvector. Instead, Google most likely uses an iterative algorithm called the Power Method to approximate the eigenvector with corresponding eigenvalue of one. Since an exact eigenvector is not needed, but an approximation is normally sufficient, relatively few iterations of the Power Method are required. This is another reason PageRank is so fast.

Even though the PageRank of the websites in the simple web structures in the assignment could easily be computing using MATLAB's "eig" command, I taught my students the Power Method and showed them how to implement it in MATLAB. I wanted the students to get a feel for how many iterations it takes to get a good estimate of the eigenvector with corresponding eigenvalue 1.

6. Concluding Remarks

For Linear Algebra instruction there are many examples of how the topic is used in many different applications. Including some material on PageRank is an excellent way to connect the idea of an eigenvector and eigenvalue to something that the students use every day. Also, since the PageRank concept is quite simple, it does not require a lot of class time to derive.

References

[1] S Brin and L Page. The anatomy of a large-scale hypertextual web search. http://infolab.stanford.edu/pub/papers/google.pdf.

[2] K Bryan and T Leise. The $25,000,000,000 Eigenvector: The Linear Algebra behind Google. *SIAM Review*, 48:569–581, 2006.

[3] A Hill. Google inside out. *Maximum PC*, pages 44–48, April 2004.

[4] A Langville and C Meyer. *Google's PageRank and Beyond: the Science of Search Engines.* Princeton University Press, 2006.

[5] C Moler. The world's largest matrix computation. http://www.mathworks.com/company/newsletters/news_notes/clevescorner/oct02_cleve.html, (accessed July 5, 2011).